

# Introduction to microarray data analysis

by Aron Eklund / [eklund@cbs.dtu.dk](mailto:eklund@cbs.dtu.dk) / February 2015

These exercises will introduce a few key topics in microarray data analysis, including the  $\log_2$  ratio, the  $t$  test, and hierarchical clustering. Basic familiarity with R is assumed.

We will use a publicly available microarray data set that was obtained from the GEO database. This data resulted from a study to find blood-based gene expression markers of Huntington's disease (HD) progression. You can read more about the data (and also perform some web-based analysis) here:

<http://www.ncbi.nlm.nih.gov/sites/GDSbrowser?acc=GDS1331>

Copy the commands given in the exercise below into the R console. Even though it is possible to simply copy and paste without much thought, I recommend that you try to understand how the functions work.

## Preliminaries

In this exercise, will use the “Biobase”, “genefilter”, and “hgu133a.db” packages from Bioconductor. If you have not yet installed Bioconductor on your computer, you should start R and type the following into the R console:

```
source("http://bioconductor.org/biocLite.R")
biocLite(c("Biobase", "genefilter", "hgu133a.db"))
```

This will first download the installer script, and then run it to install a small set of R packages that help with microarray analysis. If it asks whether you want to update any other packages, you can answer "n" (for "no").

If you want to learn more about Bioconductor in general, you can go here:

<http://www.bioconductor.org/>

## Part A: Getting familiar with the data

First you will load the data into R, and briefly examine it:

1. Start a fresh R session.
2. Check that your R workspace is empty:

```
ls()
```

This will return “character(0)” if the workspace is empty. Actually, it isn't necessary for your workspace to be empty. However, starting with an empty workspace may help you

avoid confusion. If your workspace is not empty, this probably means you have saved previous work. You can empty your workspace by typing “rm(list = ls())”.

3. Load the Biobase package, which includes (among other things) functions for working with ExpressionSet objects:

```
library(Biobase)
```

4. Load the data for this exercise directly from the CBS server:

```
load(url("http://www.cbs.dtu.dk/chipcourse/Exercises/Ex_Class/gds1331.RData"))
```

Note: Alternatively, you could first download the data to your machine, and then load it from your local disk.

5. Confirm that you have successfully loaded the data:

```
ls()
```

You should now see an object called “gds1331” present in your workspace. This object is an “ExpressionSet” that your instructor previously created using the GEOquery package.

6. Examine the data:

```
gds1331
```

Typing the name of an ExpressionSet object returns a brief summary. Under the “phenoData” section, then under the “varLabels” subsection, you can see what data is available for each patient. We are interested in the “disease.state”. You can access the phenoData slots using the dollar sign, like this:

```
gds1331$disease.state
```

The table() function can save you a bit of time counting:

```
table(gds1331$disease.state)
```

How many patients have Huntington's disease? (count both presymptomatic and symptomatic)

7. Look at the first few rows of the expression matrix:

```
head(exprs(gds1331))
```

This displays the first 6 rows of the expression matrix, where each row corresponds to a different gene (labeled by probe set ID), and each column corresponds to a different patient. The values in the matrix are gene expression values (MAS5) for each combination of sample and gene.

Question: What do the functions “head” and “exprs” do?

## Part B: Assess MBNL1 for differential expression

For this exercise, you will define two groups of patients: to distinguish Normal ("N", not HD) patients from patients with HD (either presymptomatic or symptomatic).

1. First, create a vector defining which group each patient is in:

```
group <- factor(ifelse(gds1331$disease.state == "normal",  
  yes = "N", no = "HD"))
```

Do you understand what the "ifelse()" function does? If not, you can look at its help page!

Have a look at the vector you just created:

```
group
```

2. How many patients from each group do you have?

```
table(group)
```

3. The MBNL1 gene is represented by the probe set “201152\_s\_at”. Make a plot showing expression levels of the gene MBNL1 in the two groups:

```
stripchart(exprs(gds1331)["201152_s_at", ] ~ group,  
  vertical = TRUE, method = 'jitter')
```

4. Calculate the fold change for this gene:

```
meanHD <- mean(exprs(gds1331)["201152_s_at", group == "HD"])  
meanN <- mean(exprs(gds1331)["201152_s_at", group == "N"])  
fc <- meanHD / meanN  
fc
```

5. Calculate the log<sub>2</sub> ratio for this gene:

```
log2(fc)
```

6. Calculate the  $t$  test  $P$  value for this gene:

```
t.test(exprs(gds1331)["201152_s_at", ] ~ group)
```

Is this gene differentially expressed between the two groups?

## Part C: Assess HTT for differential expression

“Huntingtin” is the gene that causes Huntington’s disease. The probe ID for huntingtin (HTT) is 202390\_s\_at.

What is its fold change, log ratio, and  $P$  value? What can you say about its differential expression?

## Part D: Clustering and heatmap (optional)

The entire data matrix is too big to visualize. Thus, we will create a more manageable sub-matrix that includes only the probe sets that are differentially expressed between normal and disease. Also, we will work with log-transformed data, to emphasize the relative changes in expression rather than the absolute values.

1. First, identify a few differentially expressed probe sets:

We must load the “genefilter” library, which contains the “rowttests” function, and the “hgu133a.db” library, which tells us which gene each probe set on the microarray detects:

```
library(genefilter)
library(hgu133a.db)
```

Calculate log ratios and  $p$ -values for each probe set:

```
tt <- rowttests(log2(exprs(gds1331)), group)
```

Create a volcano plot to visualize the  $t$ -tests on all 22283 probe sets:

```
plot(tt$dm, -log10(tt$p.value), xlab = "log2 ratio", ylab = "-log10 p-value")
```

We will (somewhat arbitrarily) select probe sets with a 2-fold change in expression and a  $P$ -value less than 0.000001

```
selected <- abs(tt$dm) > 2 & tt$p.value < 0.000001  
table(selected)
```

2. Create a sub-matrix with the most variable probe sets:

```
d <- log2(exprs(gds1331))[selected, ]
```

3. Plot a heatmap:

```
heatmap(d)
```

4. But the labels are not very informative, so we will replace the sample identifiers and probe set identifiers with disease class and gene symbols, respectively:

Create a copy of the original sub-matrix:

```
d2 <- d
```

Rename the row and column names:

```
rownames(d2) <- unlist(mget(rownames(d2), hgu133aSYMBOL))  
colnames(d2) <- group
```

5. Plot the new heatmap:

```
heatmap(d2)
```

The clustered heatmap such as this is a common way to visualize gene expression data.

6. But what do the colors mean? If you want a heatmap with a color key, you could try using the function “heatmap.2”, which can be found in the package “gplots” (which you would probably need to install).